RDF

Charlie Abela
Department of Artificial Intelligence
charlie.abela@um.edu.mt
Last Lecture

- Introduced XPath and XQuery as languages that allow for accessing and extracting node information from XML
- Problems?
Lecture Outline

- Introduction to RDF
  - Basic building blocks: resources
  - Different notations for RDF
    - Triples
    - Graphs
    - RDF/XML
  - Reification
  - Some examples

- Exercises
Moving up the layers
Scenario

- Jim’s searching experience over the Web is not brilliant
- It gets really frustrating at times when he needs to find relevant information

AARRRGHHH....
Scenario (2)

- Jim’s colleague John has recently started using an application called PEA (Personal Assistant)
- PEA can learn about the user’s interests and is able to help out John in many tasks.
- During its learning process PEA implicitly gathers information about John’s interests

- John is interested in music
- John keeps a favourites’ list of song titles
Scenario (3)

- After some time observing John’s behaviour, PEA has gathered some facts:
  - John has a list of favourite songs
  - List of favourite songs is ranked according to John’s preferences
  - John’s favourite song is “I Feel Good”
  - James Brown sings “I Feel Good”
  - “I Feel Good” is a song
  - A song has lyrics
  - John is an instance of a Person
  - James Brown is an instance of a Person
  - “I Feel Good” video is found at http://www.youtube.com/watch?v=IWcNiebYGuo
Scenario (4)

http://www.favsongs.com/singers
#sng023

http://www.singerInfo.com/singers
#singer

"James Brown"

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

http://www.w3.org/2001/vcard-rdf/3.0#name

"Mr. Dynamite"

http://www.w3.org/2001/vcard-rdf/3.0#nickname

http://www.songInfo.org/songs#title

"I Feel Good"

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

http://www.funky-stuff.com/jamesbrown/Lyrics/IFeelGood.htm

http://www.songInfo.com/songs#video

http://www.youtube.com/watch?v=IWcNiebYGuo

http://www.songInfo.com/songs#lyricsurl

http://www.songInfo.com/songs#song

http://www.singerInfo.com/singers#interprets
RDF Introduction

- Resource Description Framework is a language whose main intent is to provide a common framework for expressing machine-processable information.
- Through RDF it is possible to represent both physically accessible entities, as well as abstract entities (i.e. those that cannot be retrieved):
  - physical entity: a person’s Web page
  - abstract entity: the resource pointing to information about the person himself
- It also facilitates information exchange between different applications.
RDF: The Basics

- Basic building block: `<object attribute value>` triple
  - It is called a **statement**
  - Sentence:
    - “Dan Brown is the author of the Da Vinci Code” is such a statement
      - Object: **Dan Brown**
      - Attribute: **authorOf**
      - Value: **Da Vinci Code**
- RDF has been given an XML syntax
  - This syntax inherits the benefits of XML
  - Other syntactic representations of RDF are possible
    - N3 (Notation 3): short hand, however less popular then XML version
The fundamental components of RDF are:

- Resources: anything defined through a URI
  - `http://www.category.com/authors#Dan_Brown`
- Properties: resources that describe a relation
  - `books:authorOf` *(authorOf relation defined in books domain)*
  - essentially binary relations
- Statements: assign a value to a property associated with a specific resource
  - `<#auth1102 books:authorOf "Da Vinci Code">`
Resources

- We can think of a resource as an object, a “thing” we want to talk about (i.e. has an identity)
  - E.g. authors, books, publishers, places, people, hotels
- Every resource (physical or abstract) has a URI, a Universal Resource Identifier
- A URI can be
  - a URL (Web address)
  - some other kind of unique identifier
    - #book102 (references a unique book resource)
Properties

- Properties are special kinds of resources
- They describe relations between resources
  - E.g. “written_by”, “age”, “title”, etc.
- Properties are also identified by URIs
  - authorOf relation used in association with a book resource
    
    <books:authorOf rdf:resource="#book102"/>
RDF statements assert the properties of resources

Values can be **resources** or **literals**
- Literals are atomic values (strings)

Three views of an RDF statement
- as a triple
- as a directed graph
- as RDF/XML serialisation
Example: As a Triple

The triple \((x,P,y)\) can be considered as a logical formula \(P(x,y)\)

- Binary predicate \(P\) relates object \(x\) to object \(y\)
- RDF offers only binary predicates (properties)
- Any n-ary relation in RDF has to be converted into a set of binary relations
Example: As a Graph

- A directed graph with labeled nodes and arcs
  - **from** the resource (the **subject** of the statement)
  - **to** the value (the **object** of the statement)
- Known in AI as a *semantic net*

![Graph Diagram]

- http://www.bookstore.com/books#auth1102
- http://www.authorsInfo.com/authors#name
- http://www.booksInfo.com/books#authorOf
- "Dan Brown"
- "Da Vinci Code"
An RDF document is represented by an XML element with the tag `rdf:RDF`.

The content of this element is a number of descriptions, which use `rdf:Description` tags.

Every description makes a statement about a resource, usually identified in 2 ways:
- an `about` attribute, referencing an existing description
- an `ID` attribute, creating a new description

```xml
<rdf:Description rdf:about="#auth1102"/>
```
The `rdf:Description` element makes a statement about the resource `auth1102`.

Within the description:
- the property is used as a tag
- the content is the value of the property

```xml
<authors:name>Dan Brown</authors:name>
```
Example: RDF/XML

```xml
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:books="http://www.booksInfo.com/books#"
    xmlns:authors="http://www.authorsInfo.org/authors#">

    <rdf:Description rdf:ID="auth1102">
        <authors:name>Dan Brown</authors:name>
        <books:authorOf rdf:resource="#book102"/>
    </rdf:Description>

</rdf:RDF>
```
Concise representation

<!DOCTYPE rdf:RDF [  
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">  
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">  
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">  
<!ENTITY books "http://www.booksInfo.org/books#">  
<!ENTITY authors "http://www.authorsInfo.org/authors#">  
]>  

<rdf:RDF  
 xmlns:rdf ="&rdf;"  
 xmlns:rdfs="&rdfs;"  
 xmlns:xsd="&xsd;"  
 xmlns:authors="&authors;"  
 xmlns:books="&books;">  

 <rdf:Description rdf:ID="auth1102">  
   <authors:name>DanBrown</authors:name>  
   <books:authorOf rdf:resource="#book102"/>  
 </rdf:Description>
XML Schema data types

- In RDF, typed literals can be used
  - Usually through the use of XML Schema
- In which case the XML Schema namespace has to be declared in the namespace block

```xml
<rdf:Description rdf:ID="auth1102">
  <authors:name rdf:datatype="&xsd;string">Dan Brown</authors:name>
  <books:authorOf rdf:resource="#book102"/>
  <authors:dob rdf:datatype="&xsd;string">22/06/1964</authors:dob>
</rdf:Description>
```
Using `rdf:type`

- Declare a statement which formally defines a description

```xml
<rdf:Description rdf:ID="auth1102">
  <rdf:type rdf:resource="\&authors;author"/>
  <authors:name rdf:datatype="\&xsd;string">Dan Brown</authors:name>
  <books:authorOf rdf:resource="#book102"/>
  <authors:dob rdf:datatype="\&xsd;string">22/06/1964</authors:dob>
</rdf:Description>
```

- Declares `auth1102` to be of type `author` (itself defined in the authors vocabulary)
<rdf:RDF
    xmlns:rdf="&rdf;"
    xmlns:xsd="&xsd;"
    xmlns:books="&books;"
    xmlns:authors="&authors;">

    <rdf:Description rdf:ID="book102">
        <books:title rdf:datatype="&xsd;string">Da Vinci Code</books:title>
    </rdf:Description>

    <rdf:Description rdf:ID="auth1102">
        <rdf:type rdf:resource="&authors;author"/>
        <authors:name rdf:datatype="&xsd;string">Dan Brown</authors:name>
        <books:authorOf rdf:resource="#book102"/>
        <authors:dob rdf:datatype="&xsd;string">22/06/1964</authors:dob>
    </rdf:Description>

</rdf:RDF>
XML Vs RDF

```xml
<book ID="book102">
  <author ID="auth1102" name="Dan Brown"/>
  <title name="Da Vinci Code"/>
</book>
```

```
<rdf:Description rdf:ID="book102">
  <books:title>Da Vinci Code</books:title>
</rdf:Description>
```

```xml
<book ID="book102">
  <author ID="auth1102" name="Dan Brown"/>
  <title name="Da Vinci Code"/>
</book>
```

```
<rdf:Description rdf:ID="auth1102">
  <authors:name>Dan Brown</authors:name>
  <books:authorOf rdf:resource="#book102"/>
</rdf:Description>
```
XML vs RDF II

**Book**

- **ID**: book102
- **Author**: auth1102
  - **ID**: auth1102
  - **Name**: Dan Brown
- **Title**: Da Vinci Code
- **ISBN**: 978-0385504201
XML vs RDF III

http://www.bookstore.com/books#book102

http://www.authorsInfo.org/authors#authorOf

http://www.booksInfo.com/books#title


http://www.authorsInfo.org/authors#dob

http://www.authorsInfo.org/authors#name

22/06/64

Dan Brown

authors:author

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

Da Vinci Code

978-0385504201
Container Elements

- Collections of resources or attributes can be defined within a container element.
- RDF provides three types of containers:
  - `rdf:Bag` an unordered container, may contain multiple occurrences
  - `rdf:Seq` an order container, may contain multiple occurrences
  - `rdf:Alt` a set of alternatives
Dan Brown is the author of three books (defined elsewhere), order of publication is not important

```xml
<rdf:Description rdf:ID="auth1102">
  <authors:name>Dan Brown</authors:name>
  <books:authorOf>
    <rdf:Bag>
      <rdf:li rdf:resource="#book106"/>
      <rdf:li rdf:resource="#book102"/>
      <rdf:li rdf:resource="#book103"/>
    </rdf:Bag>
  </books:authorOf>
</rdf:Description>
```
However if its needed to specify that *authorOf* specifies ordering information (e.g. based on publication dates)

```xml
<rdf:Description rdf:ID="auth1102">
  <authors:name>Dan Brown</authors:name>
  <books:authorOf>
    <rdf:Seq>
      <rdf:li rdf:resource="#book102"/>
      <rdf:li rdf:resource="#book103"/>
      <rdf:li rdf:resource="#book106"/>
    </rdf:Seq>
  </books:authorOf>
</rdf:Description>
```
Container Elements: Alt

- **Alt** implies a list of alternatives from which only one can be chosen

```xml
<rdf:Description rdf:ID="book102">
  <books:title>Da Vinci Code</books:title>
  <book:language>
    <rdf:Alt>
      <rdf:li>engdoc</rdf:li>
      <rdf:li>mtdoc</rdf:li>
    </rdf:Alt>
  </book:language>
</rdf:Description>
```
Closed container

- If we need to specify that the listed resources are **ALL** the members of the container we need to use the **List** structure

```xml
<books:authorOf>
  <rdf:List>
    <rdf:first>
      <rdf:Description rdf:about="#book103"/>
    </rdf:first>
    <rdf:rest>
      <rdf:List>
        <rdf:first>
          <rdf:Description rdf:about="#book102"/>
        </rdf:first>
        <rdf:rest>
          <rdf:Description rdf:about="&rdf;nil"/>
        </rdf:rest>
      </rdf:List>
    </rdf:rest>
  </rdf:List>
</rdf:List>
</books:authorOf>
```
Closed collection

- A shorthand syntax for a closed collection exists

```xml
<rdf:Description rdf:ID="auth1102">
  <authors:name>Dan Brown</authors:name>
  <books:authorOf rdf:parseType="Collection">
    <rdf:Description rdf:about="#book103"/>
    <rdf:Description rdf:about="#book102"/>
    <books:authorOf>
  </rdf:Description>
</rdf:Description>
```
Reification in RDF

- In RDF it is possible to make statements about statements
  <rdf:Description rdf:ID="auth1102">
    <authors:name rdf:datatype="&xsd;string">DanBrown</authors:name>
  </rdf:Description>

- Reifies as:
  <rdf:Statement rdf:ID="auth102_S1">
    <rdf:subject rdf:resource="#auth1102"/>
    <rdf:predicate rdf:resource="&authors;name"/>
    <rdf:object rdf:datatype="&xsd;string">DanBrown</rdf:object>
  </rdf:Statement>

- The four statements are sometimes referred to as a "reification quad" for this reason.
Use of Reification

- Reification can be used to declare the provenance of a particular statement
- E.g.: suppose that for the bookstore catalogue it's important to verify who entered the data about authors
- Through reification it's possible to say:

```xml
<rdf:Description rdf:ID="john_01">
  <dc:creator rdf:resource="#auth102_S1"/>
</rdf:Description>

<rdf:Statement rdf:ID="auth102_S1">
  <rdf:subject rdf:resource="#auth1102"/>
  <rdf:predicate rdf:resource="&authors;name"/>
  <rdf:object rdf:datatype="&xsd;string">DanBrown</rdf:object>
</rdf:Statement>
```
Reification Example Graph (2)
Reification Problems

- To reify one RDF statement, four additional statements are needed.
- Leads to an explosion in the number of RDF statements.
Use of RDF: Describe Bookmarks

```xml
<rdf:Description rdf:nodeID="bk001">
  <rdf:type rdf:resource="&bookmarks;bookmark"/>
  <bookmarks:url rdf:resource="http://news.google.com/"/>
  <bookmarks:date rdf:datatype="&xsd;date">12/12/07</bookmarks:date>
  <bookmarks:query rdf:datatype="&xsd;string">news + google</bookmarks:query>
  <bookmarks:title rdf:datatype="&xsd;string">Google News</bookmarks:title>
  <bookmarks:category rdf:datatype="&xsd;string">News</bookmarks:category>
</rdf:Description>
```

`rdf:nodeID` is used to indicate an anonymous node
Use of RDF: Describe a Process

<rdf:Description rdf:nodeID="proces001">
    <rdf:type rdf:resource="&process;sysprocess"/>
    <process:name rdf:datatype="&xsd;string">lsass</process:name>
    <process:ext rdf:datatype="&xsd;string">exe</process:ext>
    <process:folder>system32</process:folder>
    <process:version>5.1.2600.2180</process:version>
    <process:company>Microsoft Corporation</process:company>
</rdf:Description>
Exercises

- Write the following statements in RDF and draw the associated graphs (use any required namespaces)
  - The library is located in Bremen
    - The book with title “Artificial Intelligence” and ISBN 978-123458 is written by the author Henry Wise
    - The author William Start wrote the books with titles “Modern Web Services” and “Theory of Computation”
    - The email address of lecturer John Brown is john.brown@widget.org
    - Lecturer John Brown has an office in the Department of AI. The office is situated on the 2nd floor of the Computer Building.

- Write RDF statements about a catalogue of products.
  - Each catalogue has an ID and a description and consists of a list of products
  - Each product is associated with a department and has a product ID and a description
  - Multiple products may be associated with the same department